INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT AKURDI, PUNE

Documentation On

# Emergency Rescue Operation

PG-DAC March 2023

**Submitted By:**

**Group No: 22**

| **Roll No.** | **Name:** |
|---|---|
| **233080** | **Shubham Sandhan** |
| **233038** | **Omkar Goud** |

**Mrs. Megha Mane**

**Project Guide**

**Mr. Rohit Puranik**

**Centre Coordinator**

# Table of Contents

## ABSTRACT

The Emergency Rescue Operation System (EROS) is a web-based application designed for efficient management of emergency incidents. It allows users to report incidents, categorize them, allocate resources, monitor incident progress, and communicate in real-time. EROS employs a Spring Boot backend, ReactJS frontend, and MySQL database. Key features include user management, incident reporting, real-time notifications, resource allocation, incident tracking, and reporting.

EROS encompasses several pivotal functionalities, including comprehensive user management, enabling account registration and access based on distinct roles. Users can promptly report incidents, providing essential details such as location and severity while attaching media files to augment incident reporting.

Moreover, EROS empowers users by categorizing incidents into predefined categories, thereby facilitating prioritization and resource allocation for optimized emergency response. Real-time notifications are disseminated through various channels, ensuring swift incident updates reach relevant parties. Emergency responders can engage in immediate messaging for effective collaboration.

Resource management is another critical facet of EROS, enabling the allocation of personnel, equipment, and supplies with visibility into availability and quantities. Users can track incident statuses and monitor progress, while emergency responders can update incident statuses and offer essential comments.

## ACKNOWLEDGEMENT

We would like to express our sincere gratitude to everyone who has contributed to the completion of our project.

First and foremost, We would like to thank our project guide **Mrs. Megha Ma'am** for their constant guidance and support throughout the project. We extend our sincere thanks to our respected Centre Co-Ordinator **Mr. Rohit Puranik**, for allowing us to use the facilities available.

We would also like to express our appreciation to the faculty members of our department for their constructive feedback and encouragement. Their insights and suggestions have helped us to refine our ideas and enhance the quality of our work.

Furthermore, we would like to thank our families and friends for their unwavering support and encouragement throughout our academic journey. Their love and support have been a constant source of motivation and inspiration for us.

Thank you all for your valuable contributions to our project.

**Shubham Sandhan (233080)**

**Omkar Goud (233038)**

# INTRODUCTION

In a world where emergencies and disasters can strike unexpectedly, having a reliable system for managing these situations is crucial. The Emergency Rescue Operation System, or EROS, is a web-based application designed to help people handle emergencies better.

This document outlines what EROS can do and what's needed to build it. It's like a blueprint for creating a powerful tool that will make it easier for authorities and emergency responders to manage crises and accidents.

EROS uses modern technology, like Spring Boot and ReactJS, to make it user-friendly and accessible to everyone. It simplifies tasks like reporting incidents, sorting them into categories, and communicating quickly during emergencies.

This document will explain how EROS manages users, helps report incidents, categorizes them, and keeps everyone informed in real-time. It also ensures the system is secure, reliable, and easy to use.

EROS is here to make emergencies easier to handle, so communities and organizations can keep people safe and minimize damage when disaster strikes.

## Features: -

1. **Administrator:** Manages user accounts, resources, and system configuration.
2. **Emergency Responder**: Reports incidents, allocates resources, and updates incident statuses.
3. **Regular User:** Reports incidents, receives notifications, and views incident updates.

## 1.1 PROJECT OBJECTIVE

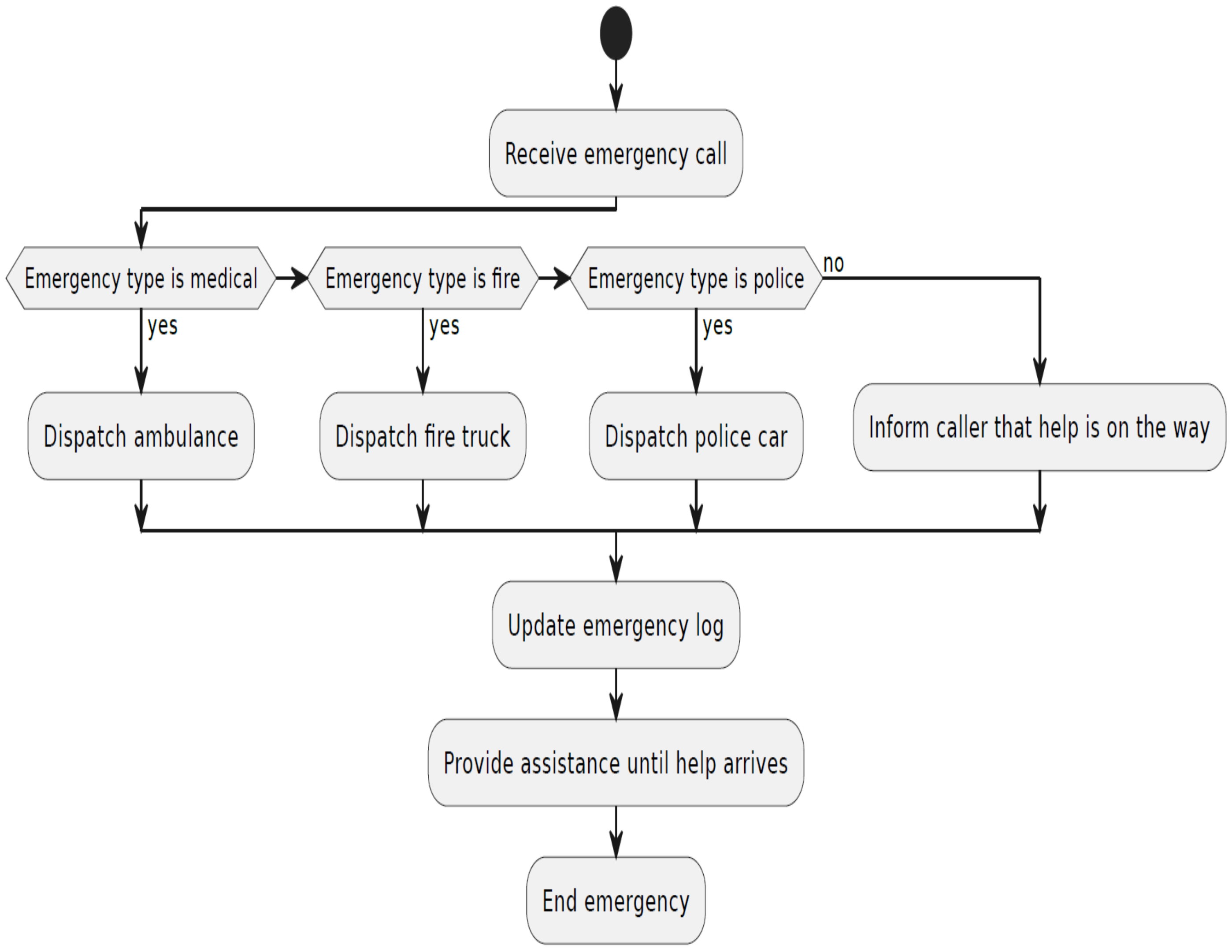
The purpose of the Emergency Rescue Operation System is to provide a digital platform that enables efficient and effective management of emergency incidents, resources, and communication during various types of disasters or crises and Accident.

### 1.2 PROJECT SCOPE

The system will allow users to report incidents, categorize them, allocate resources, monitor incident progress, and communicate in real-time. It will be accessible through web and, ensuring user-friendly interaction and responsiveness.

## Activity Diagrams:

**Activity Diagram**


Receive emergency call
Emergency type is medical
yes
Dispatch ambulance
Emergency type is fire
yes
Dispatch fire truck
Emergency type is police
yes
Dispatch police car
no
Inform caller that help is on the way
Update emergency log
Provide assistance until help arrives
End emergency

# 2.Software Requirements Specification for Emergency Rescue Operation

**1. Introduction**
**1.1 Purpose**
The purpose of the Emergency Rescue Operation System is to provide a digital platform that enables efficient and effective management of emergency incidents, resources, and communication during various types of disasters or crises and Accident.
**1.2 Scope**
The system will allow users to report incidents, categorize them, allocate resources, monitor incident progress, and communicate in real-time. It will be accessible through web and, ensuring user-friendly interaction and responsiveness.
**1.3 Overview of the SRS**
This document outlines the functional and non-functional requirements of the Emergency Rescue Operation System. It describes the system's features, interactions, and constraints to guide the development process.
**2. System Overview**
**2.1 Description of the System**
The Emergency Rescue Operation System is a web-based application built on the Spring Boot framework for the backend and ReactJS for the frontend. It allows users to report incidents, allocate resources, track incident progress, and receive real-time notifications. The system supports different user roles, including administrators, emergency responders, and regular users.
**2.2 High-Level Architecture**
The system consists of a multi-tier architecture:
Presentation Layer: ReactJS-based frontend for user interaction.

Application Layer: Spring Boot-based backend for business logic and API management.

Data Layer: MySQL database for storing incident, user, resource, and notification data.

**2.3 User Roles and Responsibilities**
Administrator: Manages user accounts, resources, and system configuration.

Emergency Responder: Reports incidents, allocates resources, and updates incident statuses.

Regular User: Reports incidents, receives notifications, and views incident updates.

**2.4 Interaction between Components/Modules**
Components interact through RESTful APIs. The frontend communicates with the backend for user authentication, incident reporting, resource allocation, and notification retrieval. **Software Requirements Specification for Emergency Rescue Operation**

## 2.5 Operating Environment

**Core java, Spring boot**
**ReactJS**
**MySQL Db**
**Rest API ,Spring Security**

## 3. Functional Requirements

### 3.1 User Management

Users can register accounts with unique usernames and passwords.

Users can log in using their credentials.

Roles and permissions determine user access levels.

Administrators can manage user accounts, including adding, editing, and deleting users.

### 3.2 Incident Reporting

Users can report incidents, providing incident details, location, and severity.

Incidents can be categorized during reporting.

Users can attach media files (photos, videos) to incidents.

### 3.3 Incident Categorization

Incidents are categorized based on predefined incident categories.

Categories assist in incident prioritization and resource allocation.

### 3.4 Communication and Notification

Users receive real-time notifications regarding incident updates.

Notifications are sent via email, SMS, or push notifications.

Emergency responders can communicate through instant messaging for collaboration.

### 3.5 Resource Management

Resources include personnel, equipment, and supplies.

Emergency responders can allocate resources to incidents.

Resources have quantities and availability status.

**Software Requirements Specification for Emergency Rescue Operation**

**Future Scope**
3.6 Incident Tracking and Status
Users can view incident statuses and progress.

Emergency responders can update incident statuses and provide comments.

3.7 Reporting and Analytics
The system generates reports on incident history, resource utilization, and response times.

## 4. Non-Functional Requirements
### 4.1 Performance
The system should handle simultaneous incident reports and updates.

Response times for critical actions should be within acceptable limits.

### *4.2 Security*

User passwords are stored securely using encryption techniques.

Role-based access control ensures proper authorization.

Data transmission is encrypted using HTTPS.

### *4.3 Reliability*
The system should have a high availability rate, with minimal downtime.

Regular data backups and disaster recovery plans are in place.

### *4.4 Usability*
The user interface should be intuitive and user-friendly.

Mobile responsiveness ensures usability on various devices.

## *5. System Constraints*
The system will be developed using Spring Boot for the backend and ReactJS for the frontend.

The database will be MySQL for data storage.

The application will be hosted on a cloud infrastructure.

# 2. SYSTEM DESIGN

System design is the solution for the creation of a new system. This phase focuses on the detailed implementation of the feasible system. Its emphasis on translating design. Specifications to performance specification. System design has two phases of development.

- Logical Design
- Physical Design

During logical design phase the analyst describes inputs (sources), outputs(destinations), databases (data stores) and procedures (data flows) all in a format that meets the user requirements. The analyst also specifies the needs of the user at a level that virtually determines the information flow in and out of the system and the data resources. Here the logical design is done through data flow diagrams and database design. The physical design is followed by physical design or coding. Physical design produces the working system by defining the design specifications, which specify exactly what the candidate system must do. The programmers write the necessary programs that accept input from the user, perform necessary processing on accepted data and produce the required report on a hard copy or display it on the screen.

## 3.1 INPUT AND OUTPUT DESIGN

### 3.1.1 INPUT DESIGN:

Input design is the link that ties the information system into the world of its users. The input design involves determining the inputs, validating the data, minimizing the data entry and provides a multi-user facility. Inaccurate inputs are the most common cause of errors in data processing. Errors entered by the data entry operators can be controlled by input design. The user-originated inputs are converted to a computer-based format in the input design. Input data are collected and organized into groups of similar data. Once identified, the appropriate input media are selected for processing. All the input data are validated and if any data violates any conditions, the user is warned by a message. If the data satisfies all the conditions, it is transferred to the appropriate tables in the database. In this project the student details are to be entered at the time of registration. A page is designed for this purpose which is user friendly and easy to use. The design is done such that users get appropriate messages when exceptions occur.

### 3.1.2 OUTPUT DESIGN:

Computer output is the most important and direct source of information to the user. Output design is a very important phase since the output needs to be in an efficient manner. Efficient and intelligible output design improves the system relationship with the user and helps in decision making. Allowing the user to view the sample screen is important because the user is the ultimate judge of the quality of output. The output module of this system is the selected notifications.

# DATABASE DESIGN

## 3.2 DATABASE

Databases are the storehouses of data used in the software systems. The data is stored in tables inside the database. Several tables are created for the manipulation of the data for the system. Two essential settings for a database are

▪ Primary key - the field that is unique for all the record occurrences

▪ Foreign key - the field used to set relation between tables Normalization is a technique to avoid redundancy in the tables.

## 3.3 SYSTEM TOOLS

The various system tools that have been used in developing both the front end and the back end of the project are being discussed in this chapter.

### 3.3.1 FRONT END:

React is a library which is developed by Facebook are utilized to implement the frontend. React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single page or mobile applications. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

### 3.3.2 BACKEND:

**Spring-Boot:**

This is used to connect MYSQL and fetch data from database and store the data in database. The Spring Framework is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE (Enterprise Edition) platform. Although the framework does not impose any specific programming model, it has become popular in the Java community as an addition to the Enterprise JavaBeans (EJB) model. The Spring Framework is Open-source Framework.

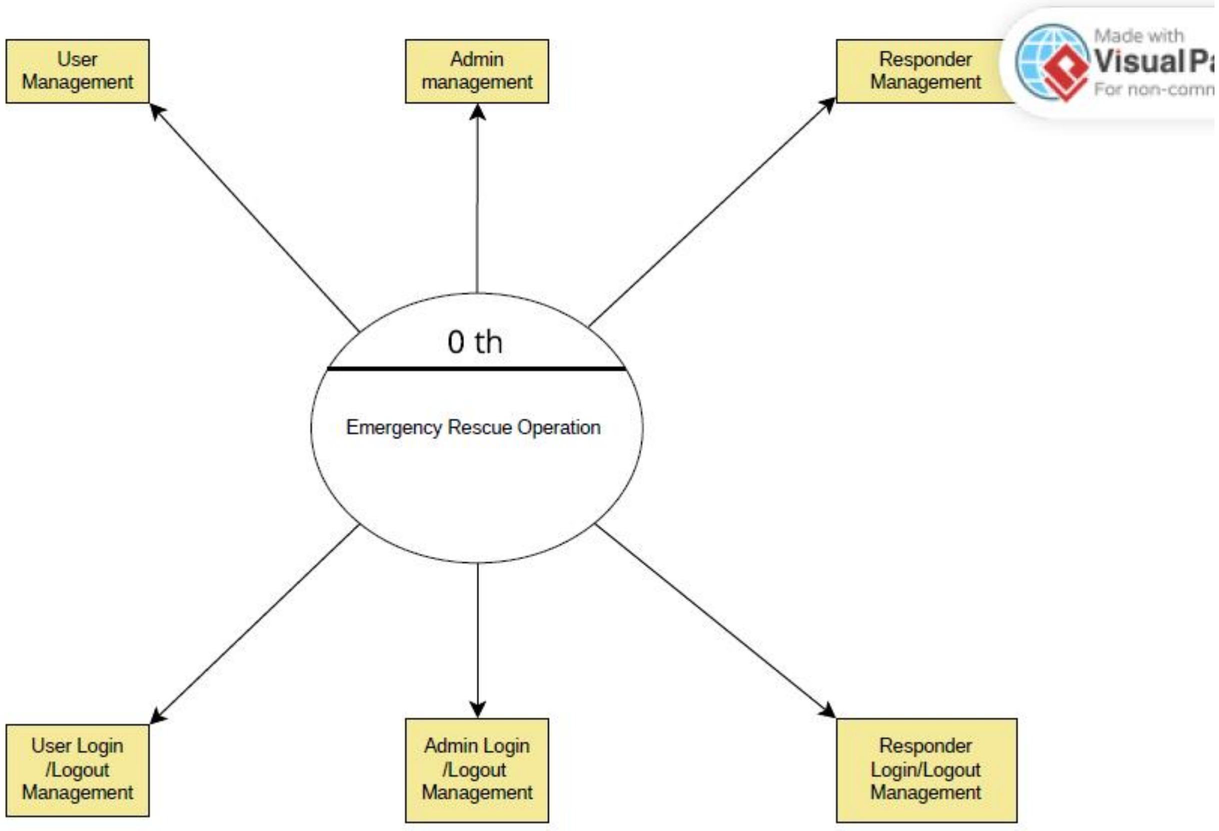
### 3.3.3 Database:

MySQL is used to design databases.

**MySQL:**
MySQL is the world's second most widely used open-source relational database management system (RDBMS). The SQL phrase stands for Structured Query Language.

## DFD Diagrams:

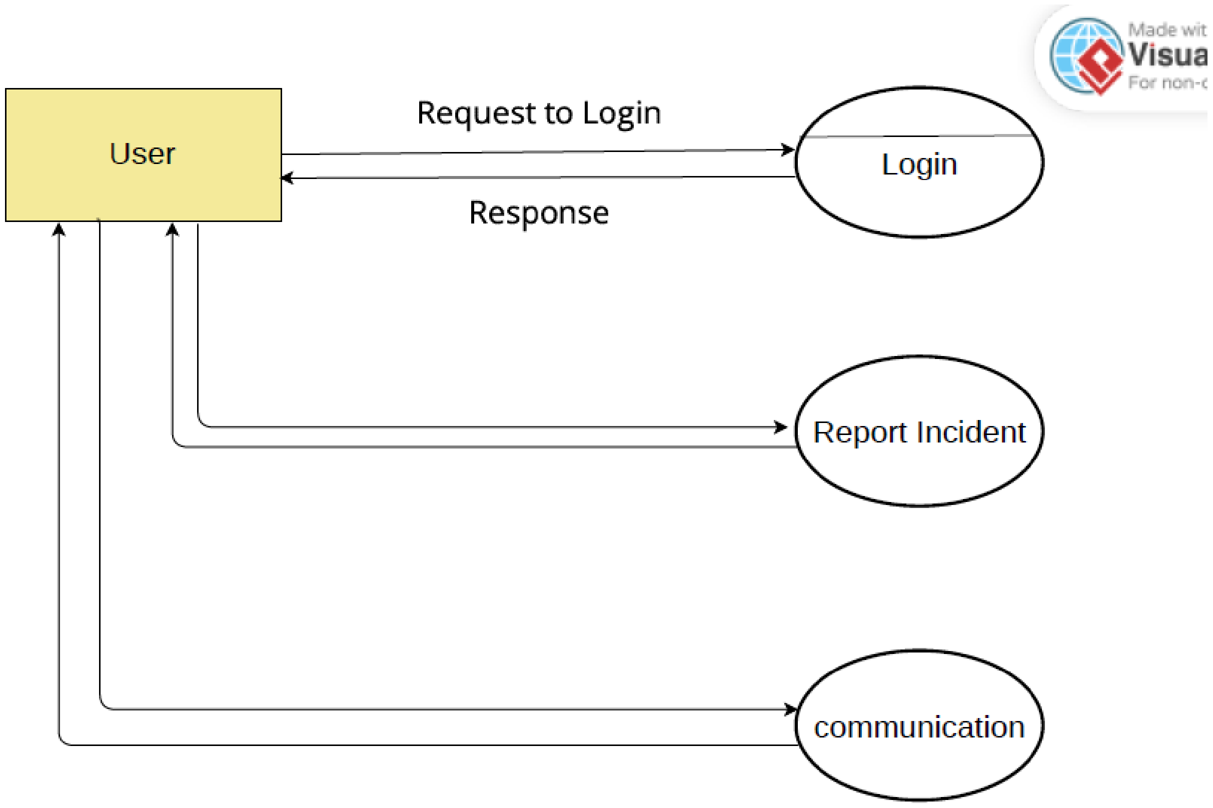
## 1 LEVEL DFD


User Management
Admin management
Responder Management
Made with Visual Pa
For non-comm
0 th
Emergency Rescue Operation
User Login /Logout Management
Admin Login /Logout Management
Responder Login/Logout Management

Zero Level DFD- Emergency Rescue Operation

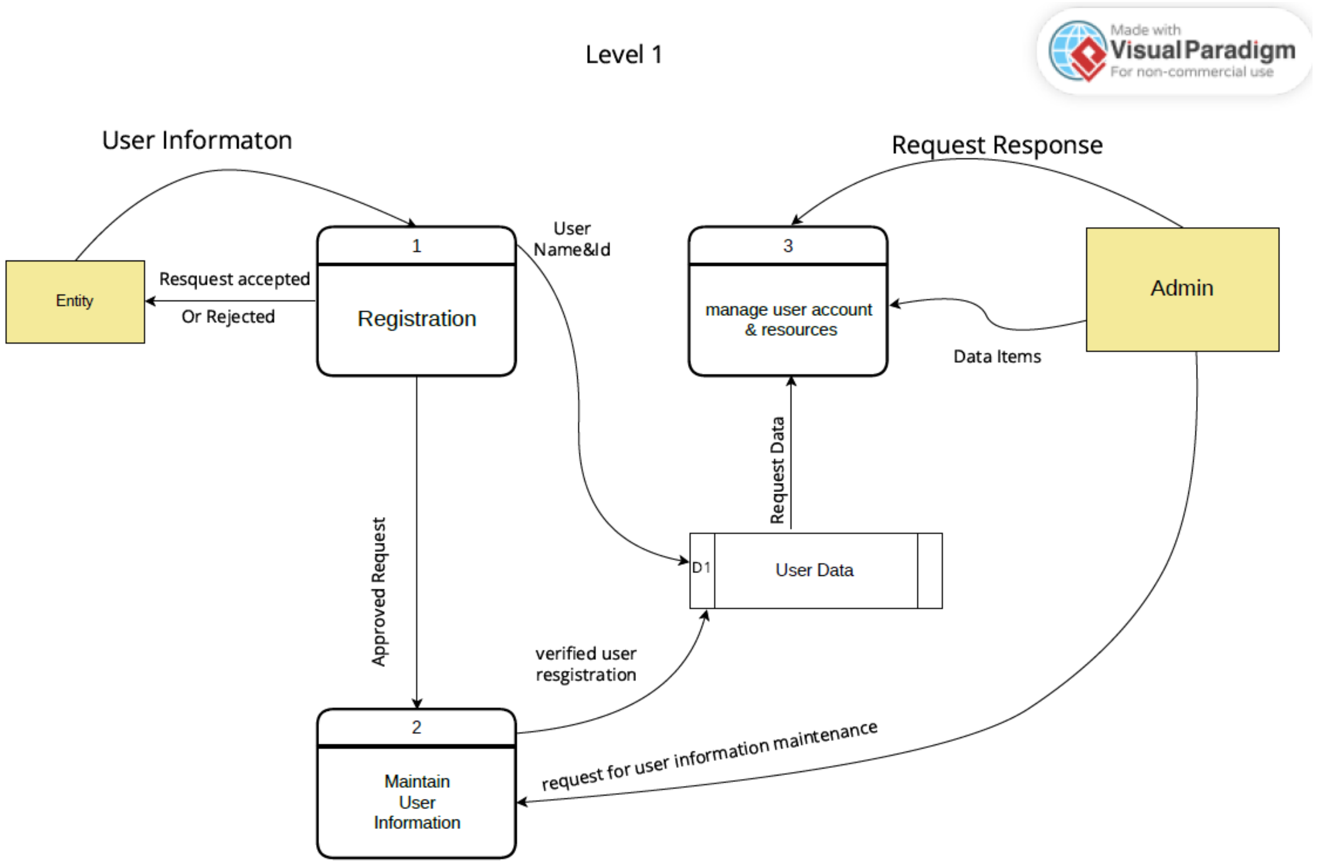
*Figure DFD Diagram*

Level 0



zero level DFD for User-Emergency Rescue Operation

Level 1


Made with Visual Paradigm For non-commercial use
User Informaton
Request Response
Entity
Resquest accepted
Or Rejected
1
Registration
User Name&Id
3
manage user account & resources
Admin
Data Items
Request Data
Approved Request
D1
User Data
verified user resgistration
2
Maintain User Information
request for user information maintenance

Level 1 DFD -Emergency Rescue Operation

## Use Case Diagram

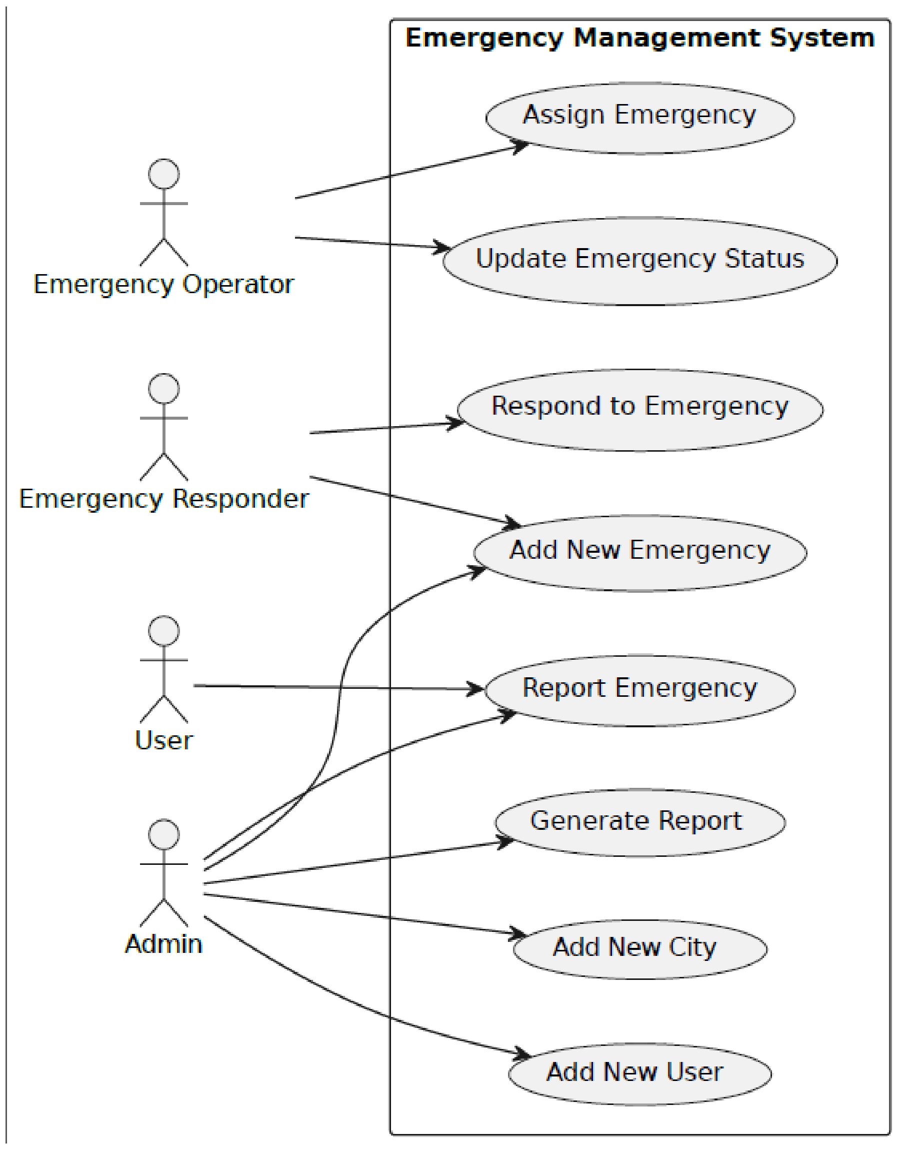


*Figure 9 Use Case Diagram*

## E-R Diagram:

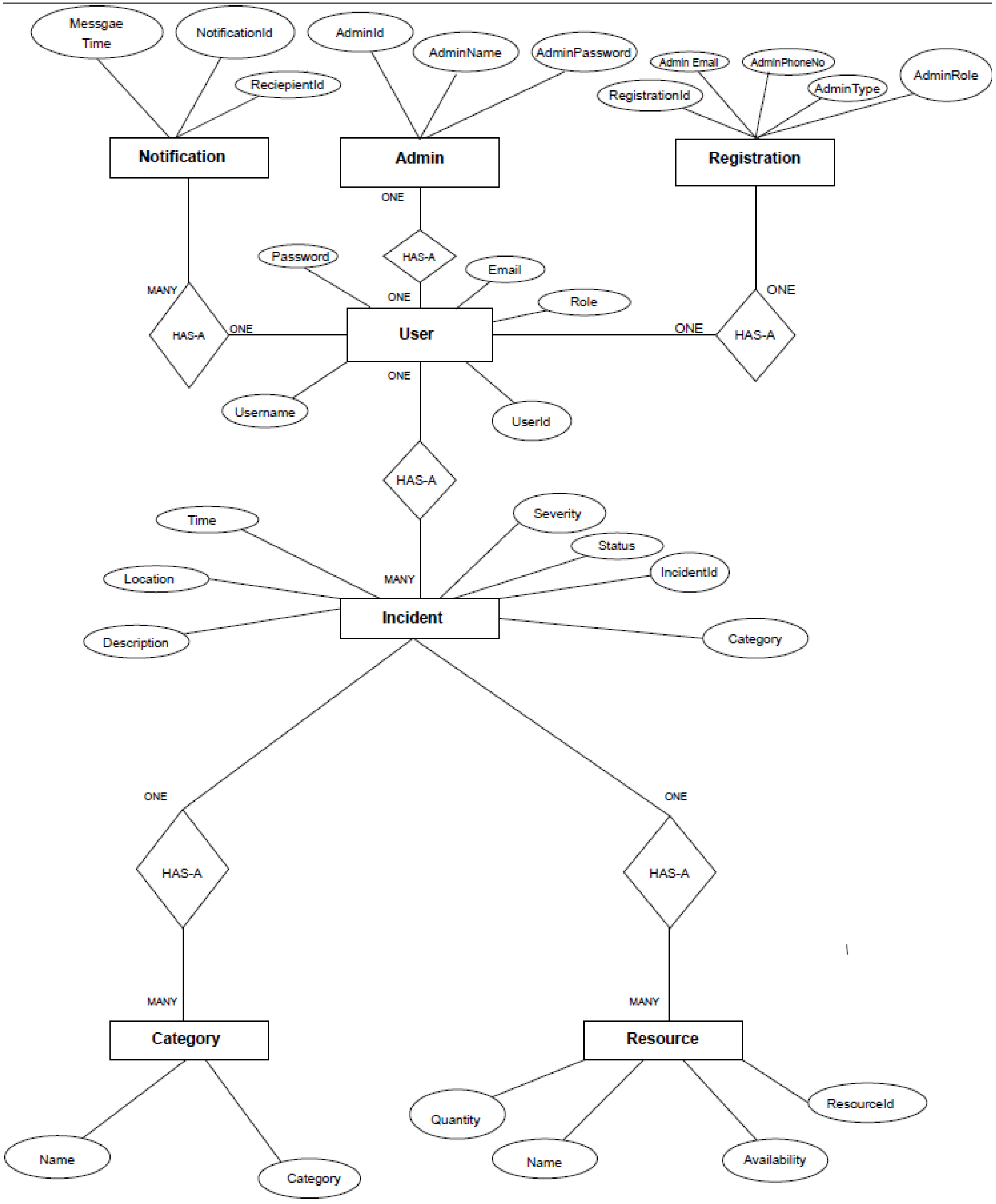
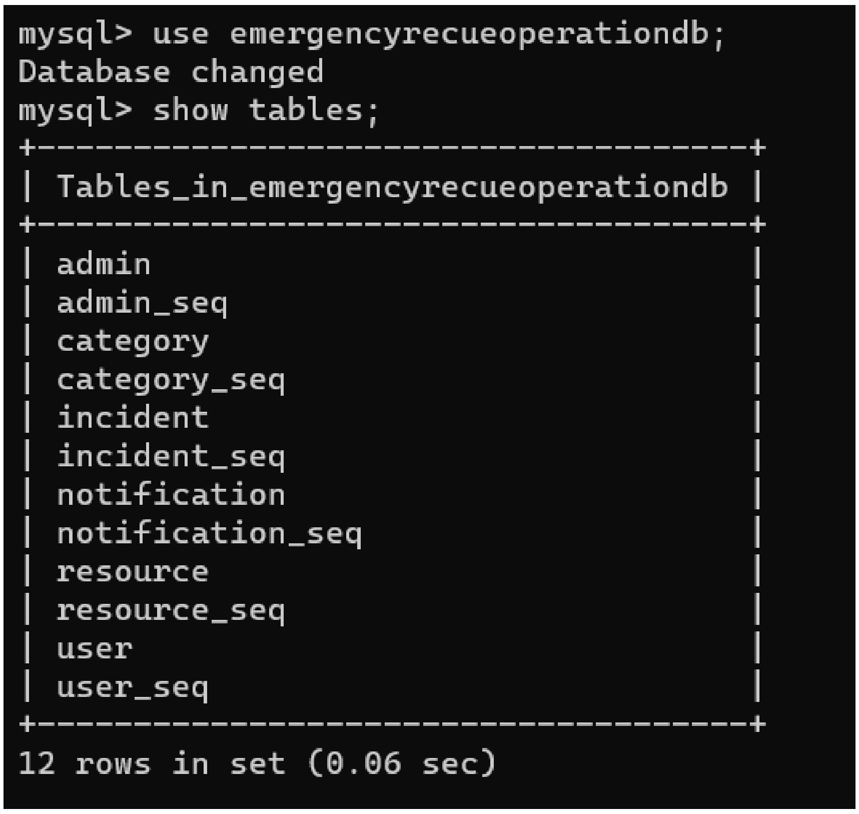


*Figure ER Diagram*

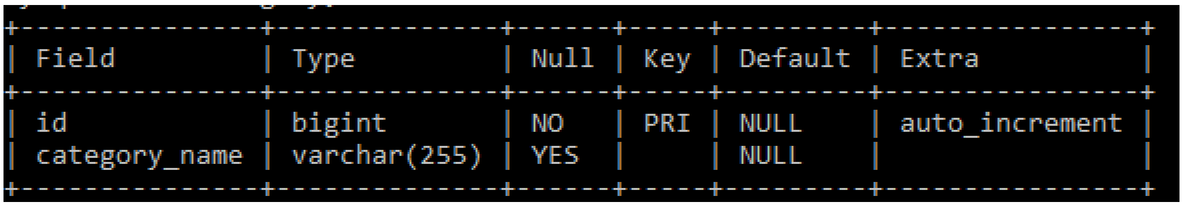
**TABLE STRUCTURE:**

**Tables:**



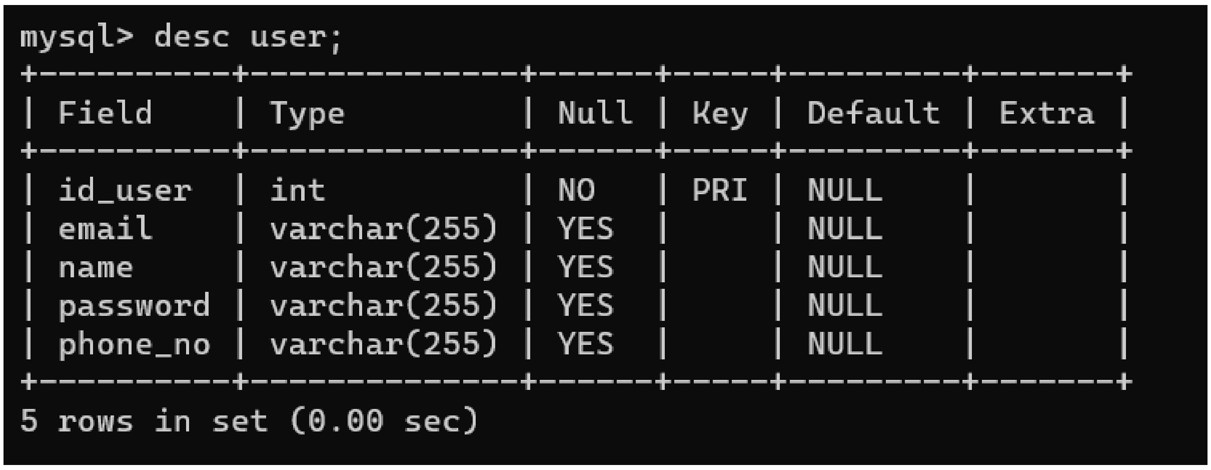
```
mysql> use emergencyrecueoperationdb;
Database changed
mysql> show tables;
+-------------------------------------+
| Tables_in_emergencyrecueoperationdb |
+-------------------------------------+
| admin                               |
| admin_seq                           |
| category                            |
| category_seq                        |
| incident                            |
| incident_seq                        |
| notification                        |
| notification_seq                    |
| resource                            |
| resource_seq                        |
| user                                |
| user_seq                            |
+-------------------------------------+
12 rows in set (0.06 sec)
```

**Category:**



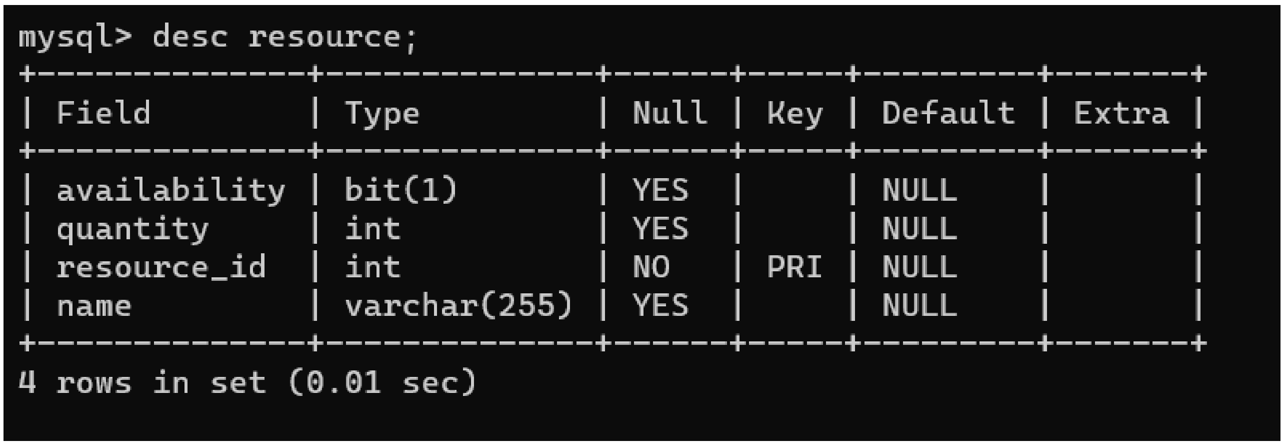
```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| id            | bigint       | NO   | PRI | NULL    | auto_increment |
| category_name | varchar(255) | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

**User:**



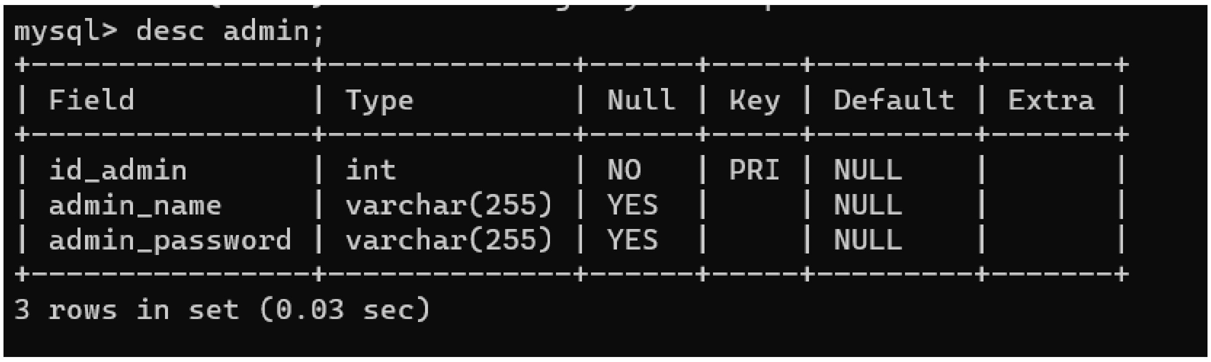
```
mysql> desc user;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| id_user  | int          | NO   | PRI | NULL    |       |
| email    | varchar(255) | YES  |     | NULL    |       |
| name     | varchar(255) | YES  |     | NULL    |       |
| password | varchar(255) | YES  |     | NULL    |       |
| phone_no | varchar(255) | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Resources:



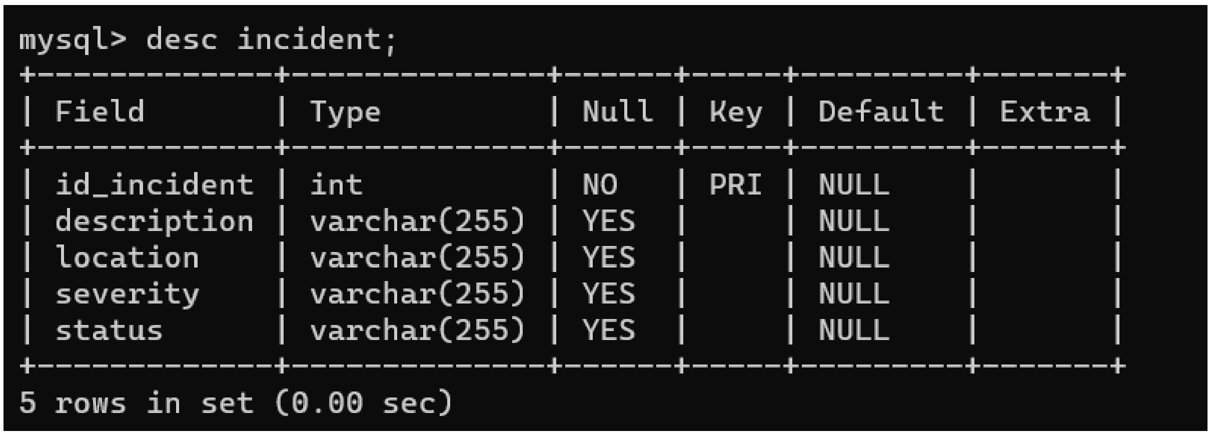
```
mysql> desc resource;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| availability | bit(1)       | YES  |     | NULL    |       |
| quantity     | int          | YES  |     | NULL    |       |
| resource_id  | int          | NO   | PRI | NULL    |       |
| name         | varchar(255) | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

**Admin:**



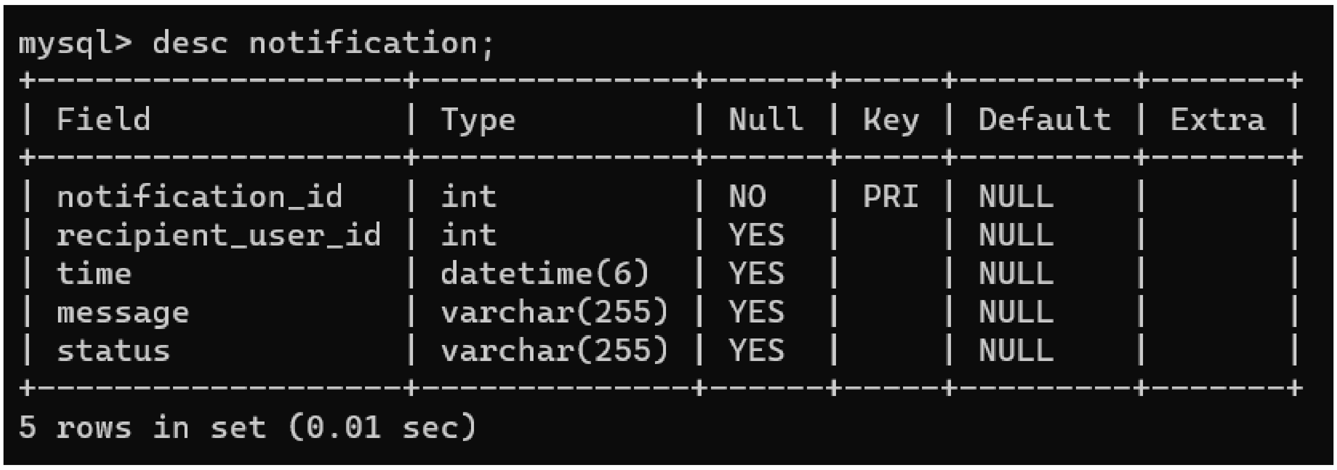
```
mysql> desc admin;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| id_admin       | int          | NO   | PRI | NULL    |       |
| admin_name     | varchar(255) | YES  |     | NULL    |       |
| admin_password | varchar(255) | YES  |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
3 rows in set (0.03 sec)
```

**Incident:**



```
mysql> desc incident;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| id_incident | int          | NO   | PRI | NULL    |       |
| description | varchar(255) | YES  |     | NULL    |       |
| location    | varchar(255) | YES  |     | NULL    |       |
| severity    | varchar(255) | YES  |     | NULL    |       |
| status      | varchar(255) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

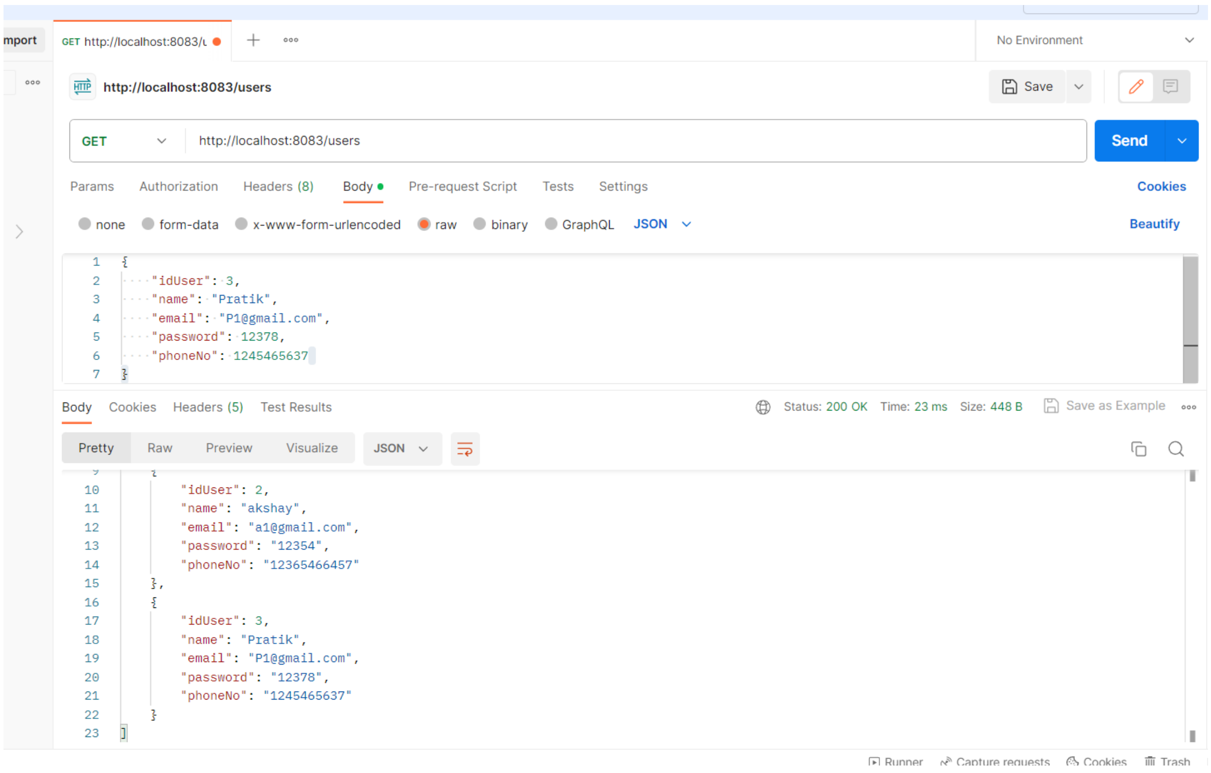
**Notification:**



```
mysql> desc notification;
+-------------------+--------------+------+-----+---------+-------+
| Field             | Type         | Null | Key | Default | Extra |
+-------------------+--------------+------+-----+---------+-------+
| notification_id   | int          | NO   | PRI | NULL    |       |
| recipient_user_id | int          | YES  |     | NULL    |       |
| time              | datetime(6)  | YES  |     | NULL    |       |
| message           | varchar(255) | YES  |     | NULL    |       |
| status            | varchar(255) | YES  |     | NULL    |       |
+-------------------+--------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

## Screenshots:

## Postman:



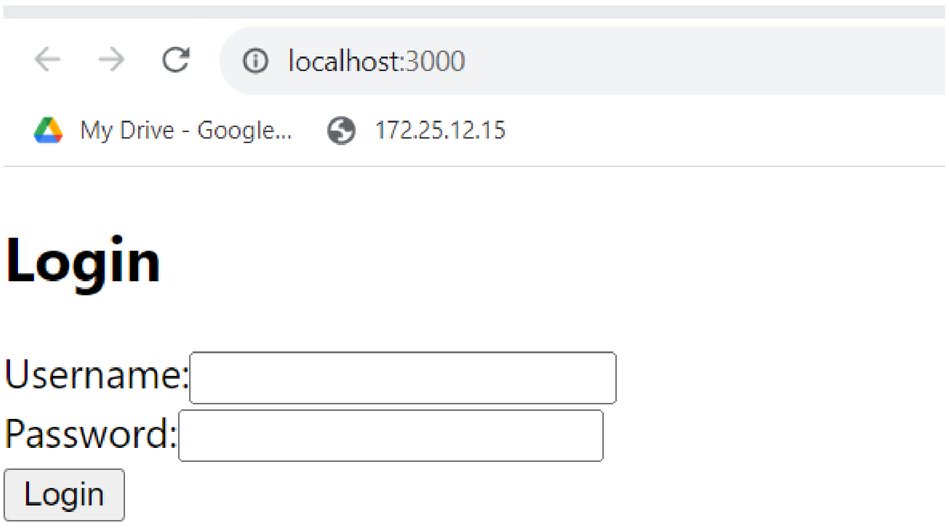
GET http://localhost:8083/users

```
{
    "idUser": 3,
    "name": "Pratik",
    "email": "P1@gmail.com",
    "password": 12378,
    "phoneNo": 1245465637
}
```

Status: 200 OK  Time: 23 ms  Size: 448 B

```
        "idUser": 2,
        "name": "akshay",
        "email": "a1@gmail.com",
        "password": "12354",
        "phoneNo": "12365466457"
    },
    {
        "idUser": 3,
        "name": "Pratik",
        "email": "P1@gmail.com",
        "password": "12378",
        "phoneNo": "1245465637"
    }
]
```

## Sign In


localhost:3000

My Drive - Google...
172.25.12.15

Login

Username:
Password:
Login

## Future Scope :

Incident Tracking and Status

- Users can view incident statuses and progress.
- Emergency responders can update incident statuses and provide comments.

Reporting and Analytics

- The system generates reports on incident history, resource utilization, and response times.

# Conclusion

In conclusion, the Emergency Rescue Operation System (EROS) presents a promising solution for efficient emergency incident management. EROS, with its user-friendly design and robust feature set, has the potential to significantly improve emergency response efforts. By simplifying incident reporting, categorization, communication, resource allocation, and data analysis, EROS aims to enhance our ability to safeguard lives and resources during critical situations. As development progresses, EROS represents a beacon of hope for more effective and resilient emergency management

**References:**

-  https://www.w3schools.com/
-  https://react-bootstrap.github.io/components/carousel/
-  https://www.geeksforgeeks.org/reactjs-tutorials/
-  https://javaee.github.io/javaee-spec/javadocs/
-  https://reactjs.org/docs/getting-started.html